In the Specification

Please amend the specification of this application as follows:

Rewrite the paragraph at page 1, lines 16 to 17 as follows:

Serial Number 09/483,697, entitled "EMULATION SUSPEND MODE WITH FRAME ~~CONTROLLED RESOURCE~~ ACCESS " claiming priority from U.S. Provisional Application No. 60/120,809 filed ~~February 19, 1999,~~ now U.S. Patent No. 6,643,803;--

Rewrite the paragraph at page 1, lines 23 to 25 as follows:

--Serial Number 09/483,783, entitled "EMULATION SYSTEM WITH ADDRESS COMPARISON UNIT AND DATA COMPARISON UNIT OWNERSHIP ARBITRATION" claiming priority from U.S. Provisional Application No. 60/120,791 filed February 19, 1999, now U.S. Patent No. 6,606,590; and --

Rewrite the paragraph at page 7, line 20 to page 8, line 14 as follows:

--Access adapter 2 is a combination of hardware and software that connects the debug host computer 1 to target system 3. Access adapter 2 utilizes one or more hardware interfaces and/or protocols to convert messages created by user interface commands of debug host computer 1 into debug commands operable on target system 3. Access adapter 2 can be either loosely coupled or tightly coupled to the debug host computer 1. In the case of a PC host computer, access adapter ~~3~~ 2 can be an XDS 510 scan controller attached directly to the PC bus. This implements a tightly coupled configuration requiring the PC to perform even the lowest level actions necessary to manage debug activity. In loosely coupled configurations, debug host computer 1 CPU communicates with another processor over a high bandwidth interface such as a SCSI, LAN or other interface. An example of this is a XDS 510WS controller

connected to the target system debug interface and to the PC through a SCSI port. In this case access adapter 2 is a debug subsystem manager and handles the detailed manipulation of the target debug capability, and debug host computer 1 send high level commands to the debug subsystem. Access adapter 2 returns data and error conditions to debug host computer 1. Current PC operating systems may not be able to service the low level debug requirements continuously. Thus it may be advantageous to partition the problem into the display and user interface and control sections.--

Rewrite the paragraph at page 11, lines 3 to 16 as follows:

--Figure 4 illustrates an electrical connection view of the coupling between access adapter 2 and target system 3. Figure 4 shows the connections of the of the various signals of the JTAG header 5 illustrated in Figure 2. All these signals are connected to scan controller 41. The signals nTRST, TCK and TMS are connected to two example megamodules ~~31~~ 45 and ~~33~~ 47. Figure 4 illustrates the optional connection of TCKO to the target system clock SYSCLK. The scan input TDI connects to a scan input of megamodule ~~31~~ 45. The scan output of megamodule ~~31~~ 45supplies the scan input of megamodule ~~33~~ 47. The scan output of megamodule ~~33~~ 47 supplies the scan output TDO. The two extension signals nET0 and nET1 control megamodules ~~31~~ 45and ~~33~~ 47 via merge unit ~~32~~ 46. These extension signals are monitored by test equipment 43.--

Rewrite the paragraph at page 16, line 31 to page 17, line 11 as follows:

--The example system in Figure 4 shows the system connectivity necessary for debug with one or more devices, one containing a programmable digital processor core. Figure 4 omits signal buffering and other electrical considerations necessary to create a functional system. In this example, target device 3 contains two

megamodules ~~31~~ 45 and ~~33~~ 47. Megamodule ~~31~~ 45 includes a programmable digital processor core while megamodule ~~33~~ 47 does not. The two devices share a parallel connection to signals nTRST, TCK, and TMS. The scan path begins as TDI at the connector, enters megamodule ~~31~~ 45, exits megamodule ~~33~~ 47, and ends as TDO back at the connector. Connections between merge unit ~~32~~ 46 and pins nET1 and nET0 create trigger channels one and zero.--

Rewrite the paragraph at page 17, line 23 to page 18, line 10 as follows:

--Figure 6 illustrates the data needed to completely load a target device. A complete target device read uses this same data format. This data would typically include a prefix section 110 corresponding to registers within the serial scan path before the desired register. The data includes the data 111 corresponding to the desired register. Lastly, the data typically includes a suffix section 113 directed to data corresponding to ~~register~~ registers following the desired register in the serial scan path. It can be quickly ~~seem~~ seen that for a single register read or write, most of the data transfer bandwidth is directed to transferring data not necessary for accessing the desired register. It is known to provide particular modules of the target device with a bypass capability. Using such a bypass capability causes the serial data stream to be diverted from the serial scan registers of that module and coupled directly from the module's serial data input to its serial data output. Even so, it is often the case for complex products that a minimum serial scan path including the desired register to include hundreds of times the number of bits as that of the desired register.--

Rewrite the paragraph at page 19, line 18 to page 20, line 2 as follows:

--Example module 200 illustrated in Figure 8 may also transit data via this alternate data transfer protocol. Programmable digital processor core 220 loads the data to be transmitted to data register OUT 212. Programmable digital processor core 220 then triggers start bit generator 213 and ~~selected a~~ selects data transmission mode at output switch 202. Start bit generator 213 produces the start bit which is selected for transmission to the TDO line by output switch 202. Output switch 202 then selects data register OUT 212 for serial transmission of the predetermined number of bits. This scheme is similar to that of a universal asynchronous receiver/transmitter (UART) with start bits and fixed length data. Note that the data transmitted to module may be data to be loaded into selected locations within the module or it may be instructions for execution by programmable digital processor core 220.--